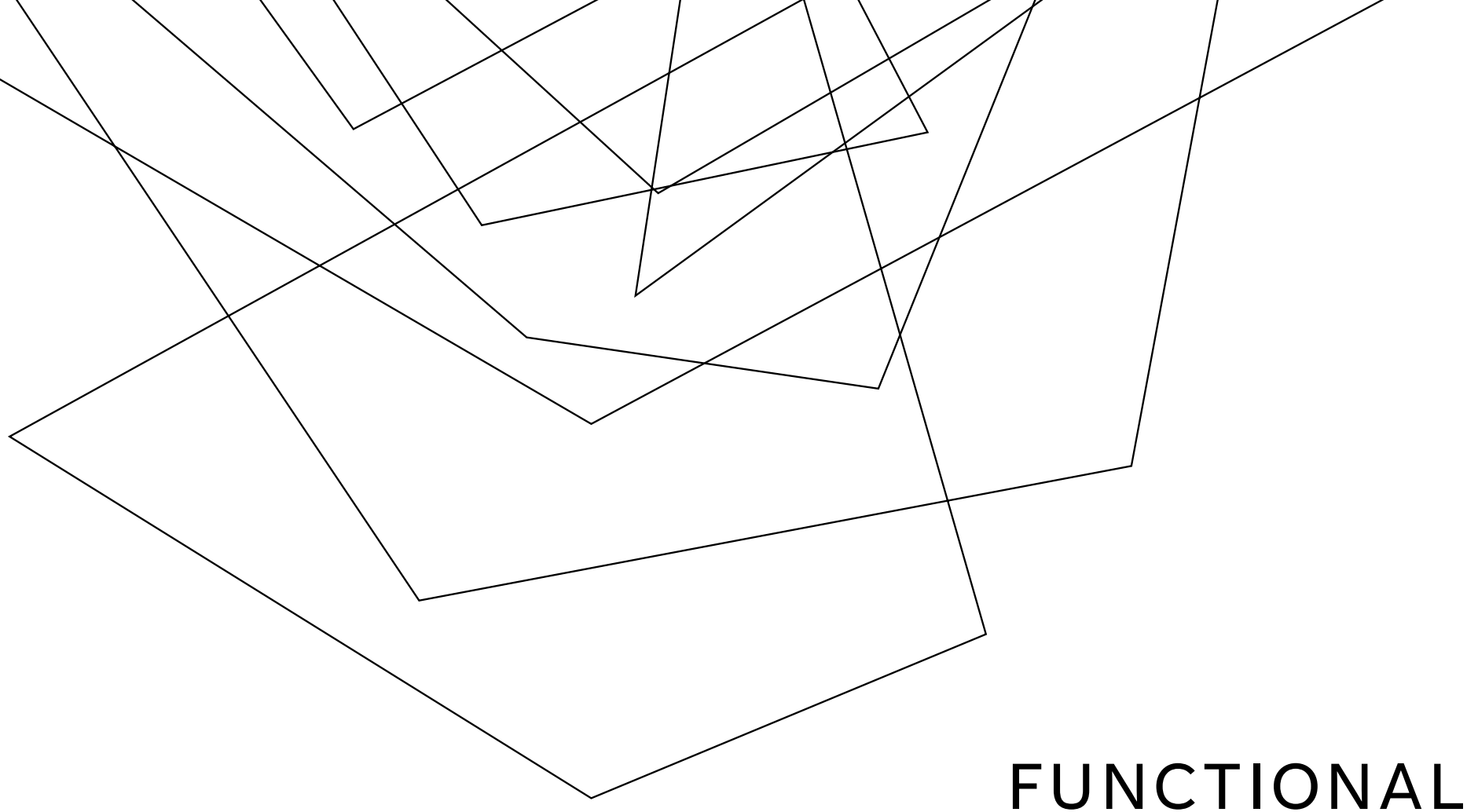


FUNCTIONAL ENCRYPTION

Sihui Wang Dec 5th, 2022

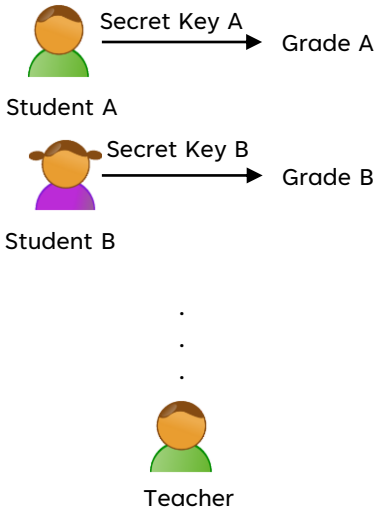
MOTIVATION AND CONTEXT

A-Prior Secret Key → Public Key Encryption → ?

Public Key Cryptography allows two parties to communicate without having an a-priori mutual secret

However ...

Possible Problem:
Suppose that each student holds a secret key to his/her own grades



Teacher wants to get access to class average. What key(s) does he/she need?

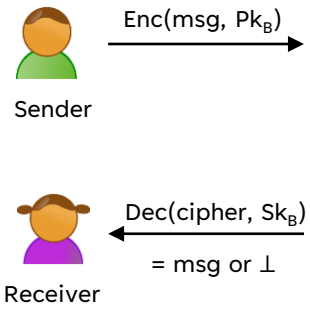
MOTIVATION AND CONTEXT

A-Prior Secret Key → Public Key Encryption → ?

Public Key Cryptography System's Assumption:

1) Encryption is a method to send a message or data to a ***single*** entity holding a secret key

2) Access to encrypted data is ***all or nothing***



"One-to-one" key pairs (Pk, Sk)
"All or nothing" access control

MOTIVATION AND CONTEXT

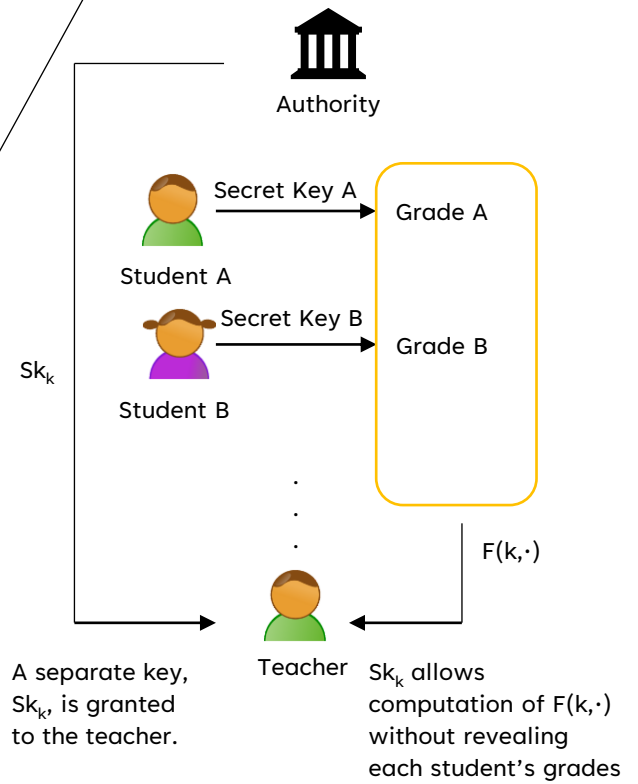
A-Prior Secret Key → Public Key Encryption → Functional Encryption

Functional Encryption:

1) An authority holding a master key can generate a key Sk_k

2) The key Sk_k grants access to a function of the message: $F(k,\cdot)$

3) Public key encryption is a special case of functional encryption.



FORMULATION OF FUNCTIONAL ENCRYPTION

A functional encryption scheme (FE)

Functionality: F; Key Space: K; Plaintext Space: X

1) Generate a public and master secret key pair
$$(pp, mk) \leftarrow setup(1^{\lambda})$$

2) Generate secret key sk for k
$$sk \leftarrow keygen(mk, k)$$

3) Encrypt message x
$$c \leftarrow enc(pp, x)$$

4) Use sk to compute F(k,x) from cipher
$$y \leftarrow dec(sk, c)$$

Require that $y = F(k, x)$ with probability 1.

Public Key Encryption:
Generate public key pp and secret key sk
Public key pp for encryption of x
Secret key sk for decryption of x

Functional Encryption:
Generate public key pp and master secret key mk
Public key pp for encryption of x
Generate secret key sk for each functionality F(k,·)
Secret key for decryption of F(k,x)

EXAMPLE: INNER PRODUCT FUNCTIONALITY UNDER DDH ASSUMPTION

DDH Assumption: (g, g^a, g^b, g^{ab}) and (g, g^a, g^b, g^c) are computationally indistinguishable

DDH Inner Product Scheme:

1) Setup: Sample from $(\mathbb{G}, p, g)$. $msk = s = (s_1, \dots, s_l)$, $mpk = (h_i = g^{s_i})_{i\in[l]}$. Return the pair (mpk, msk).

2) Encryption: choose a random r and compute $ct_0 = g^r$. Given the message $x = (x_1, \dots, x_l)$, for each $i \in [l]$, compute $ct_i = h_i^r \cdot g^{x_i}$. Return $Ct = (ct_0, (ct_i)_{i\in[l]})$.

3) Key Derivation: Given the master secret key msk and a vector $y = (y_1, \dots, y_l)$, compute the secret key $sk_y = \langle y, s \rangle$.

4) Decryption: Given mpk, Ct, sk_y, compute $\frac{\prod_{i\in[l]} ct_i^{y_i}}{ct_0^{sk_y}} = g^{\langle x,y \rangle}$

The above scheme is CPA secure under DDH assumption

BUILD FUNCTIONAL ENCRYPTION SCHEME ON TOP OF PUBLIC KEY ENCRYPTION

Functional encryption schemes can be based on any public-key encryption scheme with certain structural and homomorphic properties.

Structural Requirements:

Secret Key: From $(G, +, 0_G)$

Public Key: From $(H, \cdot, 1_H)$

Message: From $\mathbb{Z}_q$, q is a prime number

Ciphertext: has two parts Ct_0, Ct_1. Ct_0 corresponds to some commitment $C(r)$ of randomness r; Ct_1 is $Enc(pk, x; r)$.

Linear Key Homomorphism: $y_1 sk_1 + y_2 sk_2$ is the secret key to $pk_1^{y_1} pk_2^{y_2}$.

Linear Ciphertext Homomorphism Under Shared Randomness:

$$Enc(pk_1 pk_2, x_1 + x_2; r) = Enc(pk_1, x_1; r) \cdot Enc(pk_2, x_2; r)$$

INSTANTIATION WITH EL-GAMAL ENCRYPTION SCHEME

1) Key Generation:
   Choose a prime p and a primitive root g
   Choose random private keys $X_1, \dots, X_l \in \mathbb{Z}_p^*$
   Compute public keys $h_i = g^{X_i}, i \in [l]$

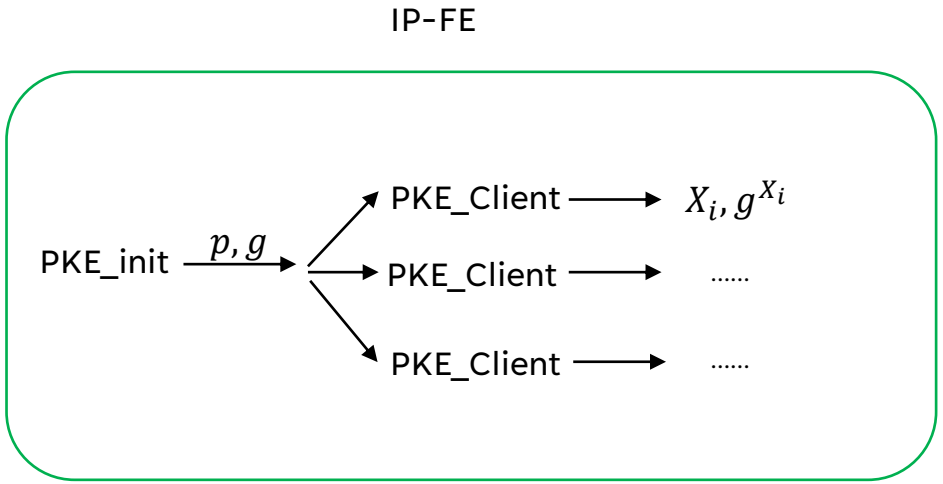
2) Key Derivation:
   For a vector $y = (y_1, \dots, y_l)$, generate secret key $Sk_y = \sum y_i X_i$.

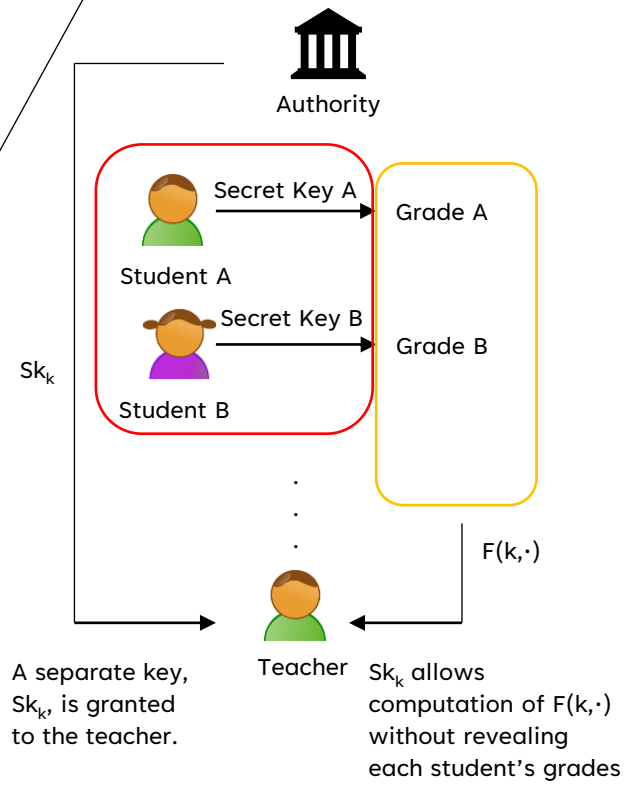
3) Encryption:
   Choose random $Y \in \mathbb{Z}_p^*$
   Compute $Ct_0 = g^Y$
   For message $x = (x_1, \dots, x_l)$, compute $Ct_i = g^{x_i} \cdot h_i^Y$

4) Decryption:
   Compute $g^{\sum x_i y_i} = \frac{\prod_{i \in [l]} ct_i^{y_i}}{ct_0^{sky}}$

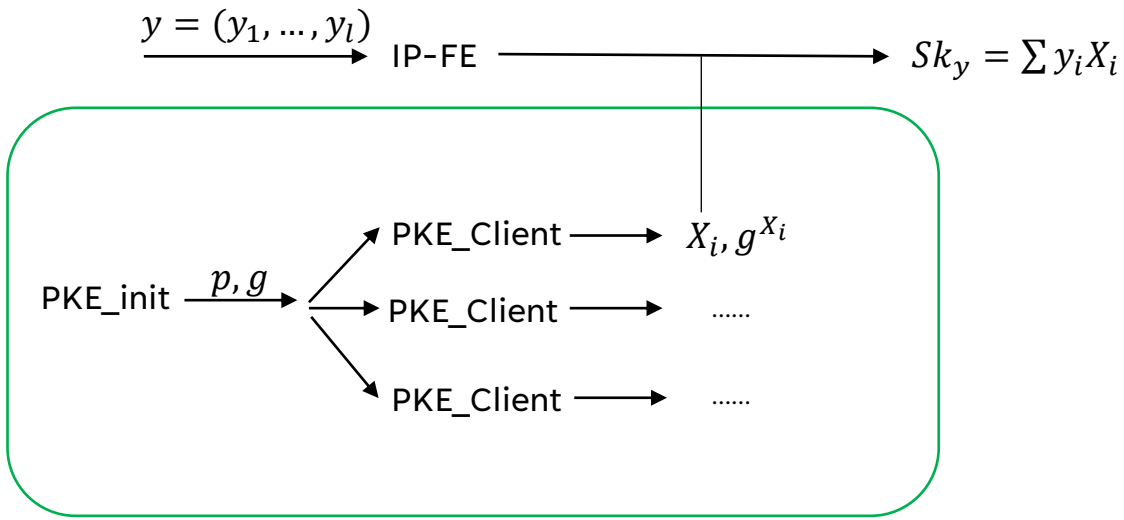
Original El Gamal:
1) Key Generation:
Choose a prime p and a primitive root g
Choose random private key $X \in \mathbb{Z}_p^*$
Compute public key $h = g^X$
2) Encryption:
Choose random $Y \in \mathbb{Z}_p^*$
Compute $Ct_0 = g^Y$ and a shared secret $s = h^Y$
Compute $Ct_1 = g^x \cdot h^Y$
Ciphertext is (Ct_0, Ct_1)
3) Decryption:
Compute shared secret $s = Ct_0^X$
Compute $g^x = \frac{Ct_1}{s}$

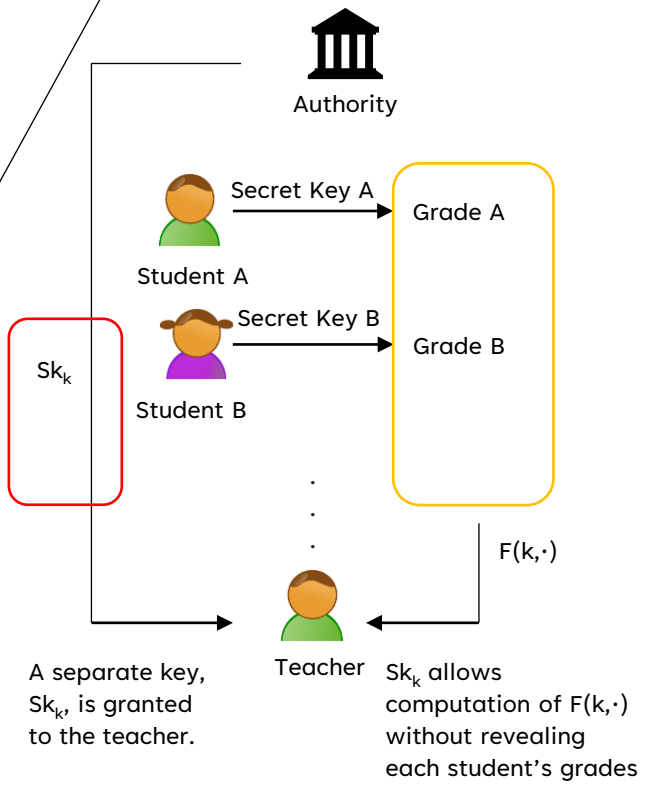
IMPLEMENTATION: KEY GENERATION



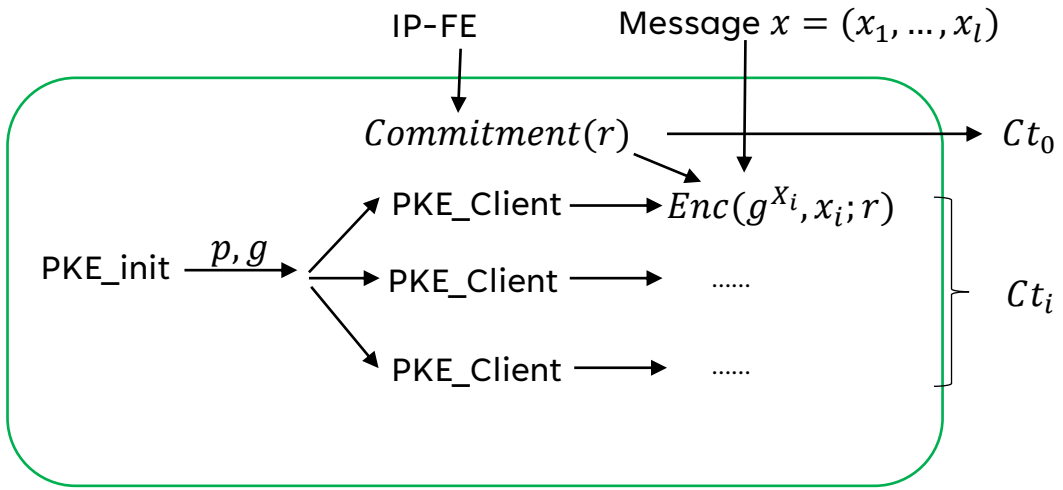


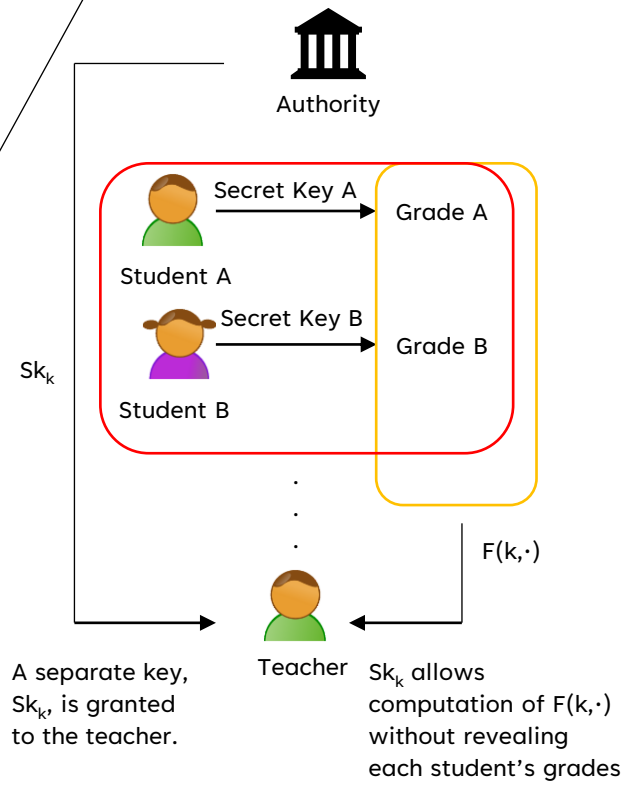
IMPLEMENTATION: KEY DERIVATION



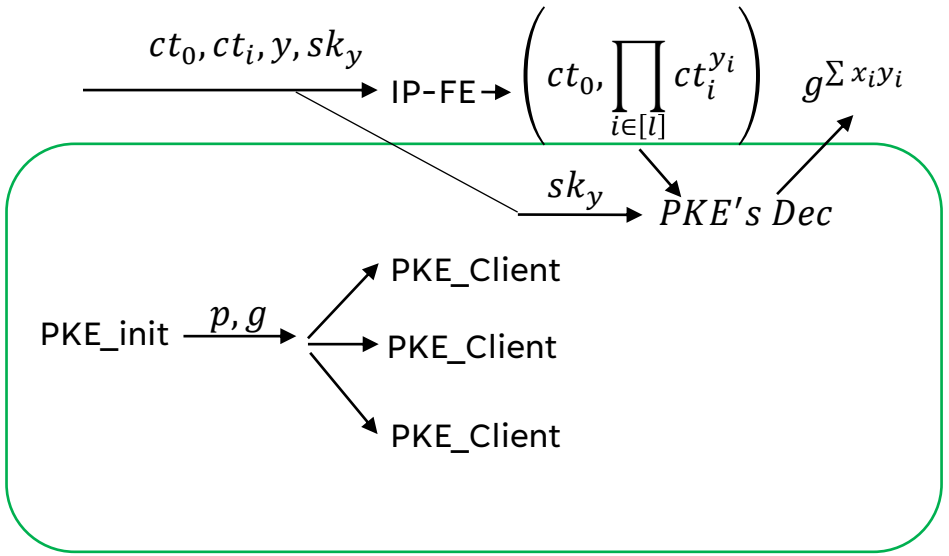


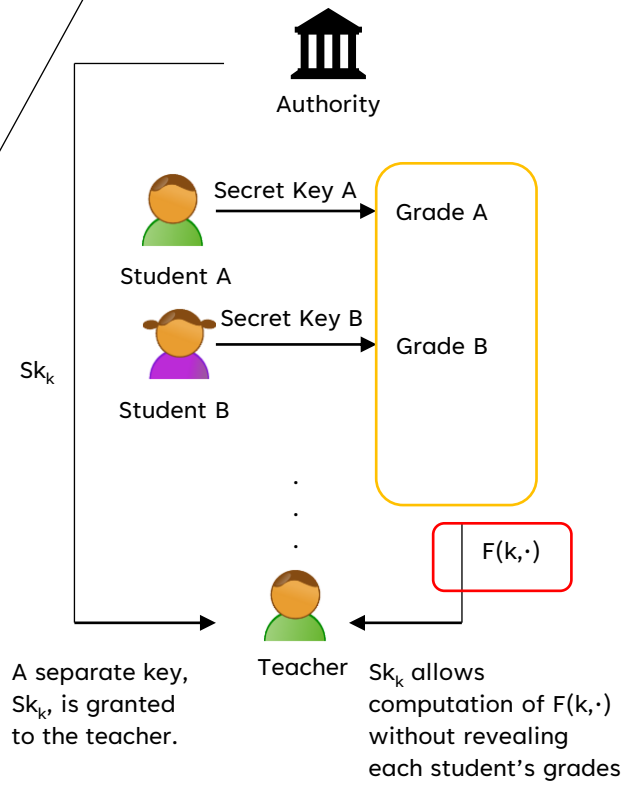
IMPLEMENTATION: ENCRYPTION





IMPLEMENTATION: DECRYPTION





SUMMARY

Compared with Public Key Encryption, functional Encryption provides fine-grained access control.

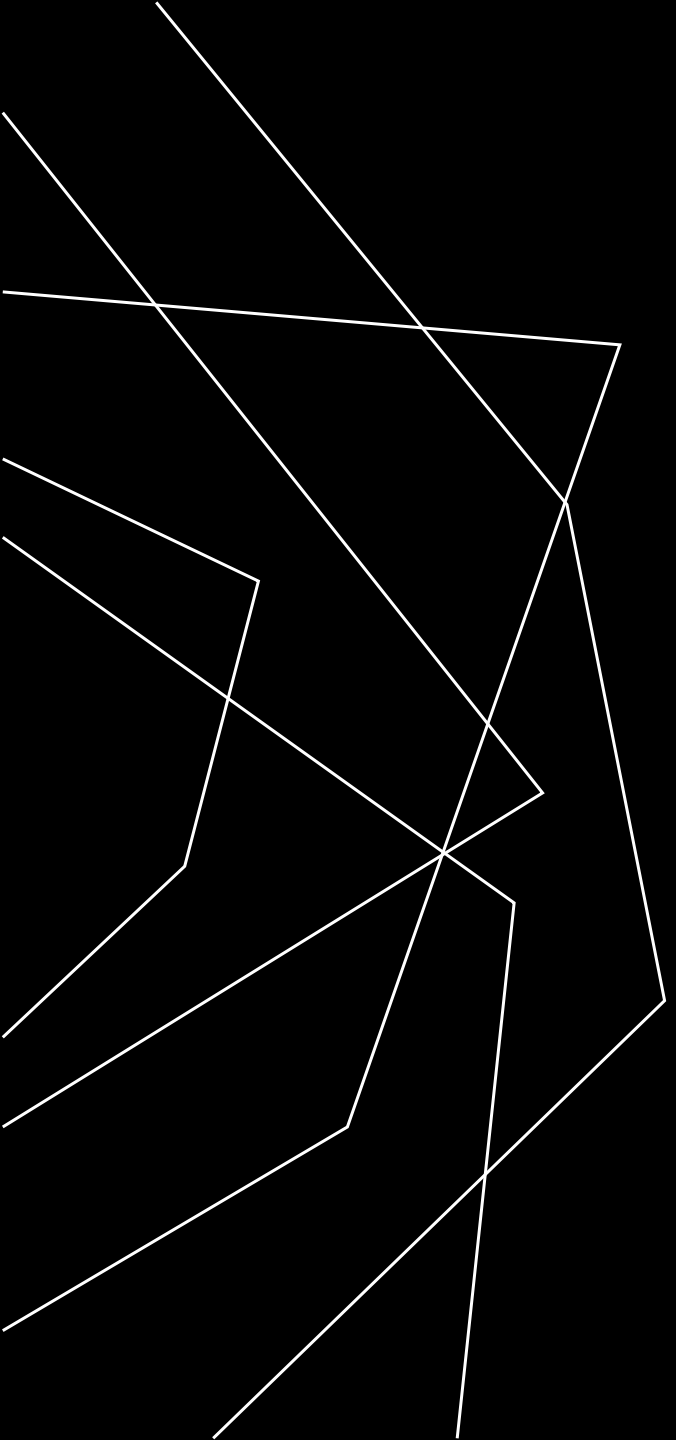
CPA secure Functional Encryption Scheme can be built on top of Public Key Encryption

El Gamal can be used to construct Functional Encryption with Inner Product functionality under DDH assumption

REFERENCE

Boneh, Dan, Amit Sahai, and Brent Waters. "Functional encryption: Definitions and challenges." In *Theory of Cryptography Conference*, pp. 253-273. Springer, Berlin, Heidelberg, 2011.

Abdalla, Michel, Florian Bourse, Angelo De Caro, and David Pointcheval. "Simple functional encryption schemes for inner products." In *IACR International Workshop on Public Key Cryptography*, pp. 733-751. Springer, Berlin, Heidelberg, 2015.



THANK YOU